Fix “Unsupported API version” when loading a Paper plugin

`Unsupported API version` indicates that the plugin declares an API version newer than the server supports. Paper's implementation uses this message when checking the plugin's declared API against the current server API. It is not the same as Java rejecting a class-file format, even if both problems appear after a plugin update.

Capture the exact rejection

Find the first loading failure for the plugin in the startup log. Record the declared API version printed in the error, the plugin release and the actual server version. Do not infer the server version from the filename alone; a file named `server.jar` can contain many releases.

If the server does not remain running long enough to use a version command, use its startup banner. Keep the full error and official download link in your notes. Redact any unrelated connection details or credentials before sending logs publicly.

Compare the compatibility statement

Read the plugin author's supported Minecraft/Paper versions for that exact artifact. Its `api-version` metadata is a compatibility declaration, not a user preference. Paper documentation explains that a server below the declared version refuses to load the plugin. Since 1.20.5, minor versions can also be included, so a broad major-release match may not be enough.

Do not confuse the API value with the Java runtime requirement. Changing Java alone cannot add a newer Minecraft API to an older server. Conversely, installing a compatible API build can still leave a separate Java class-version mismatch to resolve. Follow the first actual error at each stage rather

than changing both layers blindly.

Choose the less disruptive supported option

If the server version must stay unchanged, obtain a plugin release the author supports on that server. Preserve the previous JAR and plugin data before replacing anything. Confirm that you downloaded the Paper/Bukkit artifact rather than a proxy or mod-loader build with a similar project name.

If you genuinely need a newer server version, plan that as a separate world upgrade. Take a recoverable checkpoint, check all plugins and the Java requirement, and trial a copy before changing production. Do not upgrade an established world merely to silence one plugin warning without understanding the wider effect.

Never unpack the plugin and lower its declared API version as a hosting workaround. That can bypass a useful check while leaving calls to methods or behaviour the older server lacks. A successful load after metadata editing does not prove compatibility, and modifying the artifact makes author support harder.

Verify and retain a safe rollback

During a clean stop, install the chosen supported plugin build and start normally. Confirm the API rejection is gone, check the reported plugin version and exercise its intended function. Inspect for new method/class errors rather than declaring success because the original line disappeared.

If the plugin update changed data, follow the documented compatible rollback, not just a JAR swap. If the server itself was upgraded, do not run an older Minecraft binary against newly upgraded world data. Recovery uses the matching pre-upgrade checkpoint and a deliberate plan for any later progress. Ask support with the declared API, actual server build and official compatibility page if no supported combination is clear.

Related help

- Install plugins on a Paper or Purpur Minecraft server
- Choose the Java runtime for your Minecraft server

- Change Minecraft versions with a backup and rollback plan

Official references

Paper api-version metadata rules. Paper implementation of the API-version check.

Revision #1
Created 2026-11-09 10:55:44 UTC by CloudExa Hosting Admin
Updated 2026-11-09 10:55:44 UTC by CloudExa Hosting Admin